
Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| | Vorwort | xv |
| 1 | Vue im Überblick | 1 |
| 1.1 | Was ist ein JavaScript-Frontend-Framework? | 1 |
| 1.2 | Historie und das Team rund um Vue | 2 |
| 1.3 | Was ist Vue und was zeichnet es aus? | 3 |
| 1.4 | Warum Vue? | 3 |
| 1.5 | Unterschiede zu anderen Web-Frameworks | 4 |
| 1.6 | Vue 2 oder Vue 3 lernen? | 6 |
| 1.7 | Der Blick in die Zukunft | 7 |
| 2 | Die Grundlagen: Ein Vue-Schnellstart | 9 |
| 2.1 | Die erste Vue-Anwendung aufsetzen | 9 |
| 2.1.1 | Die HTML-Struktur | 10 |
| 2.1.2 | Einbinden von Vue als Skript | 10 |
| 2.1.3 | Unsere erste Vue-Instanz | 12 |
| 2.1.4 | Schleifen und Bedingungen | 14 |
| 2.1.5 | Benutzereingaben verarbeiten | 16 |
| 2.1.6 | Mit Komponenten kapseln | 17 |
| 2.1.7 | Das deklarative Rendering und das virtuelle DOM | 19 |
| 3 | Vue 3 im Überblick | 23 |
| 3.1 | Warum neu schreiben? | 23 |
| 3.2 | Ein mehrschrittiges Release | 24 |
| 3.3 | Interne und externe Anpassungen | 25 |
| 3.4 | Der neue Standard und Framework-Empfehlungen | 26 |

| | | |
|----------|---|-----------|
| 4 | Vue-Projekte aufsetzen | 27 |
| 4.1 | Scaffolding als Basis | 27 |
| 4.2 | Vue-Projekte erstellen | 28 |
| 4.2.1 | Das Vue CLI | 29 |
| 4.2.2 | Die Auswahlmöglichkeiten | 30 |
| 4.2.3 | Ein Projekt erzeugen und den Webserver starten | 31 |
| 4.2.4 | Rapid (instant) Prototyping | 31 |
| 4.3 | Vite als neuer Standard | 32 |
| 4.4 | Die Entwicklungsumgebung vorbereiten | 34 |
| 4.5 | Server-Side Rendering (SSR) | 35 |
| 5 | Das (neue) Reactivity-System | 37 |
| 5.1 | Der Ansatz von Vue | 39 |
| 5.2 | Die Implementierung als Proxy in Vue 3 | 41 |
| 5.3 | Probleme bei der Reaktivität | 44 |
| 6 | Komponenten im Detail | 47 |
| 6.1 | Komponentenorientierte Entwicklung | 48 |
| 6.2 | Komponenten und das Separation-of-Concerns-Prinzip | 49 |
| 6.3 | Options API, Class-Style-Syntax und Composition API | 49 |
| 6.3.1 | Options API | 50 |
| 6.3.2 | Composition API | 52 |
| 6.4 | Globale und lokale Komponenten | 52 |
| 6.5 | Single-File Components (SFC) | 54 |
| 6.6 | Lifecycle-Hooks | 57 |
| 6.7 | Daten und Props | 58 |
| 6.8 | Non-Prop-Attribute und Refs | 61 |
| 6.9 | Methoden | 62 |
| 6.10 | Benutzerdefinierte Events | 64 |
| 6.11 | Der Datenfluss zwischen Komponenten | 65 |
| 6.12 | Computed Property | 67 |
| 6.13 | Watcher | 68 |
| 6.14 | Eigene Komponenten mit v-model | 70 |
| 6.15 | Standard, Named und Scoped Slots | 72 |
| 6.16 | Dynamische Komponenten | 75 |
| 6.17 | Eigene Direktiven und Plug-ins | 76 |
| 6.18 | Fragmente, Portale und Suspense | 77 |

| | | |
|----------|---|------------|
| 6.19 | Asynchrone und funktionale Komponenten | 79 |
| 6.20 | Dependency Injection mit Provide und Inject | 80 |
| 6.21 | Render-Funktionen und JSX | 81 |
| 6.22 | Vue Components und Web Components | 82 |
| 6.23 | Wiederverwendbarkeit und Komponenten-Patterns | 83 |
| 7 | Data Binding, Formulare und Styles | 85 |
| 7.1 | Einführung ins Data Binding bei Vue | 86 |
| 7.2 | Template-Syntax und das Binding von Attributen | 88 |
| 7.3 | Bedingtes Rendering mit v-if und v-show | 89 |
| 7.4 | Rendern von Listen mit v-for | 91 |
| 7.5 | Die neue Arbeit mit Arrays | 94 |
| 7.6 | Daten filtern, suchen und sortieren | 95 |
| 7.7 | Daten auf Seiten aufteilen (Pagination) | 97 |
| 7.8 | Formulare mit Daten verknüpfen | 99 |
| 7.9 | Events von Anwenderinnen verarbeiten | 102 |
| 7.10 | Formulare validieren | 104 |
| 7.11 | Class- und Style-Binding | 106 |
| 7.12 | Animationen und Transitionen | 108 |
| 8 | Das Composition API | 113 |
| 8.1 | Der Hintergrund | 113 |
| 8.2 | Setup, Daten und Props | 116 |
| 8.3 | Script Setup | 119 |
| 8.4 | Methoden und Lifecycle-Hooks | 120 |
| 8.5 | Computed Property und Watcher | 121 |
| 8.6 | Composables (Composition Function) | 122 |
| 8.7 | Vorgefertigte Kompositionen in Bibliotheken | 125 |
| 8.8 | ref vs. reactive und Reactivity Transform | 127 |
| 8.9 | Provide und Inject | 129 |
| 8.10 | Render Functions und Templates Refs | 130 |
| 8.11 | Eine komplette Komponente mit dem Composition API ... | 132 |
| 8.12 | Das Reactivity API | 134 |
| 8.13 | Composition API vs. Options API vs. Class API | 136 |
| 8.14 | Das Composition API und Vue 2 | 137 |

| | | |
|-----------|---|------------|
| 9 | Routing mit dem Vue Router | 139 |
| 9.1 | Was ist Routing und was der Vue Router? | 139 |
| 9.2 | Installation und Bestandteile des Vue Router | 141 |
| 9.3 | Routen konfigurieren | 141 |
| 9.4 | Routen nutzen | 144 |
| 9.5 | Dynamische Routen mit Parametern und Props | 146 |
| 9.6 | Das Pattern-Matching | 150 |
| 9.7 | Die verschiedenen History-Modi | 151 |
| 9.8 | Verschachtelte Routen, Redirects und Lazy Loading | 153 |
| 9.9 | Route-Guards und Metadaten | 156 |
| 9.10 | CSS-Klassen, Transitionen und Composition API | 159 |
| 9.11 | Fehler bei der Navigation erkennen | 161 |
| 9.12 | Ein DIY-Router für Minimalistinnen und Minimalisten | 162 |
| 10 | State Management mit Vuex und Pinia | 165 |
| 10.1 | Was ist State Management und das Flux-Pattern | 165 |
| 10.2 | Andere State-Management-Patterns | 167 |
| 10.3 | Das neue Vuex und die Installation | 169 |
| 10.4 | Die drei Kernprinzipien | 170 |
| 10.5 | Die 4 + 1 Kernkonzepte | 172 |
| 10.5.1 | Der Zustand (State) | 172 |
| 10.5.2 | Zugriff auf den Zustand (Getter) | 173 |
| 10.5.3 | Veränderungen des Zustands (Mutations) | 174 |
| 10.5.4 | Aktionen durchführen (Actions) | 176 |
| 10.5.5 | Einen Store aufteilen (Module) | 177 |
| 10.6 | Einen Store in Komponenten nutzen | 178 |
| 10.7 | Vuex und das Composition API | 181 |
| 10.8 | Die Struktur einer Vuex-Anwendung und Plug-ins | 182 |
| 10.9 | Einen Store debuggen | 184 |
| 10.10 | State Management mit Pinia | 185 |
| 10.11 | Vuex bzw. Pinia nutzen oder nicht? | 188 |
| 11 | Mit Vue auf Ressourcen zugreifen | 191 |
| 11.1 | Das Beispiel-API | 191 |
| 11.2 | Zugriff auf ein API über das Fetch API | 192 |
| 11.3 | Axios vs. Fetch | 193 |

| | | |
|-----------|--|------------|
| 11.4 | Die Bibliothek Axios als Quasi-Standard | 193 |
| 11.5 | Axios in Vue integrieren | 194 |
| 11.6 | Daten abfragen und anzeigen | 196 |
| 11.7 | Daten hinzufügen und löschen mit POST und DELETE . . . | 198 |
| 11.8 | Axios konfigurieren | 199 |
| 11.9 | Pagination mit API-Aufrufen | 199 |
| 11.10 | Mit Fehlern umgehen | 200 |
| 11.11 | API-Zugriffe bündeln | 201 |
| 11.12 | Axios mit Vue Router und Vuex | 203 |
| 11.13 | Ein kurzer Blick auf Alternativen | 203 |
| 12 | Anwendungsentwicklung mit Vue | 205 |
| 12.1 | Ein Blick über den Tellerrand | 205 |
| 12.2 | Das Tooling als Grundvoraussetzung | 206 |
| 12.3 | Eine Vue-Projektstruktur | 207 |
| 12.4 | Vue-Apps debuggen und Fehlerbehandlung | 208 |
| 12.5 | Web-Apps auf Basis von UI-Frameworks | 209 |
| 12.6 | Web-Apps auf Basis von Application Frameworks | 211 |
| 12.7 | Mobile App-Entwicklung und statische Seiten | 212 |
| 12.8 | Eine Web-App am Beispiel von Quasar | 213 |
| 13 | Internationalisierung und Barrierefreiheit | 215 |
| 13.1 | Installation und Konfiguration von Vue I18n | 216 |
| 13.2 | Das aktive Gebietsschema abrufen und verändern | 218 |
| 13.3 | Übersetzungen nutzen | 219 |
| 13.4 | Vue I18n und das Composition API | 222 |
| 13.5 | Asynchrones Laden von Übersetzungsdateien | 224 |
| 13.6 | Routen und Router-Links übersetzen | 224 |
| 13.7 | Vuex und Vue I18n | 226 |
| 13.8 | Anwendungsentwicklung und das Ökosystem | 226 |
| 13.9 | Etwas zur Barrierefreiheit | 228 |
| 14 | Testen von Vue-Anwendungen | 229 |
| 14.1 | Das Testen und Vue | 230 |
| 14.2 | Vorbereitungen im Projekt | 233 |
| 14.3 | Tests organisieren und strukturieren | 233 |

| | | |
|-----------|--|------------|
| 14.4 | Unit-Tests für JavaScript und TypeScript | 234 |
| 14.5 | Unit-Tests für Vue-Komponenten | 235 |
| 14.6 | Unit-Tests für das Composition API | 238 |
| 14.7 | Events testen | 239 |
| 14.8 | Tests für Formulareingaben | 241 |
| 14.9 | Tests für Vue Router | 241 |
| 14.10 | Tests für Vuex | 243 |
| 14.11 | Ende-zu-Ende-Tests | 249 |
| 14.12 | Snapshot-Tests | 250 |
| 14.13 | Tests der Internationalisierung | 251 |
| 15 | Build und Deployment von Vue-Anwendungen | 253 |
| 15.1 | Development- vs. Production-Build | 253 |
| 15.2 | Den Build im Auge behalten | 255 |
| 15.3 | Deployment auf Netlify und Heroku | 257 |
| 15.4 | CI-/CD-Pipelines mit GitHub Actions und Azure DevOps | 257 |
| 16 | Performante Vue-Apps entwickeln | 259 |
| 16.1 | Vue 3 als allgemeiner Vorteil | 259 |
| 16.2 | Performance von Komponenten prüfen | 260 |
| 16.3 | Events und Reactivity | 261 |
| 17 | Migration von Vue 2 auf Vue 3 | 263 |
| 17.1 | Die Wege einer Migration und der Migration-Build | 264 |
| 17.2 | Breaking Changes | 265 |
| 17.2.1 | Globales API | 266 |
| 17.2.2 | Template-Direktiven | 267 |
| 17.2.3 | Das Key-Attribut | 267 |
| 17.2.4 | Komponenten | 268 |
| 17.2.5 | Render-Funktionen | 269 |
| 17.2.6 | Viele kleinere Änderungen | 269 |
| 17.3 | Vue Router und Vuex | 270 |
| 18 | Vue und TypeScript | 273 |
| 18.1 | Vue (3) und TypeScript | 273 |
| 18.2 | Ein neues Vue-Projekt mit TypeScript | 274 |
| 18.3 | TypeScript im Vue-Projekt | 275 |

| | | |
|-----------|---|------------|
| 18.4 | Die verschiedenen API-Modelle | 275 |
| 18.4.1 | Options API | 276 |
| 18.4.2 | Class-Style Vue Components | 277 |
| 18.4.3 | Composition API | 278 |
| 18.5 | Vue Router, Vuex und Internationalisierung | 280 |
| 18.6 | Vor- und Nachteile bei Vue mit TypeScript | 281 |
| 18.7 | Migration auf TypeScript | 282 |
| 19 | Vue und das Backend | 283 |
| 19.1 | Express (Node.js, JavaScript und TypeScript) | 283 |
| 19.2 | Django (Python) | 284 |
| 19.3 | Laravel (PHP) | 284 |
| 19.4 | Spring Boot (Java) | 285 |
| 19.5 | ASP.NET Core (.NET, C#, F# und Visual Basic) | 285 |
| 19.6 | Firebase (Low Code) | 286 |
| 19.7 | Serverless als Motto | 286 |
| 19.8 | Low- und No-Code als Wachstumsmotor | 287 |
| 20 | Etwas zu Bibliotheken | 289 |
| 20.1 | Composition API | 289 |
| 20.2 | Awesome Vue als umfassender Überblick | 290 |
| 20.3 | Eine Bibliothek für Vue 2 und Vue 3 erstellen | 291 |
| | Literaturverzeichnis | 293 |
| | Index | 299 |